Database

Базы данных

База данных (БД) — это имеющая название совокупность данных, которая отражает состояние объектов и их отношений в рассматриваемой предметной области.

Данными называют зарегистрированную информацию, представление фактов, понятий или инструкций в форме, которая подходит для передачи, связи, обработки человеком или с помощью машины. **Содержимое базы данных** — прайс-листы, контакты пользователей, каталоги товаров, отчеты, статистика продаж и т.д. Изменения одной ячейки автоматически влияют на другие.

Свойства базы данных

Из определения базы данных следует, что в ней:

- всегда есть имя. Если имя не задано, то нет и базы данных;
- фиксируется состояние объектов и их отношений в заданный момент времени. Со временем оно меняется. Например, цена товара может характеризовать его состояние. Вслед за изменением цены меняется и состояние товара;
- фиксируется информация об объектах из определенной предметной области. Например, если рассматриваем предметную область «Библиотека», то в базе могут фиксироваться данные по книгам, их расположению в библиотеке, читателям и читательским билетам. Если наша предметная область — «Магазин», то в БД может находиться информация по товарам и их ценам, по торговым точкам и наличию товара в конкретной торговой точке.

Важной характерной чертой БД является ее постоянство. Оно проявляется в нескольких контекстах:

- данные постоянно накапливаются и используются;

- состав и структура данных обычно постоянны и стабильны во времени. Если они меняются, то скорее всего БД находится в процессе проектирования и разработки;
- элементы данных могут меняться (вслед за изменением состояний объектов и их отношений). Тем самым информация, которую содержит каждая база данных, постоянно актуализируется.

Система управления базами данных

Система управления базами данных (СУБД) – это комплекс программно-языковых средств, позволяющих создать базы данных и управлять данными. Иными словами, СУБД — это набор программ, позволяющий организовывать, контролировать и администрировать базы данных. Большинство сайтов не могут функционировать без базы данных, поэтому СУБД используется практически повсеместно.

Основные функции СУБД:

- управление данными во внешней памяти (на дисках);
- управление данными в оперативной памяти с использованием дискового кэша;
- журнализация изменений (сохранение истории), резервное копирование и восстановление базы данных после сбоев;
- поддержка языков БД (язык определения данных, язык манипулирования данными)

Отличия БД и СУБД

Категория сравнения	СУБД	БД
Определение	— специальные приложения (или библиотеки) для управления базами данных различных размеров и форм.	*— специально разработанное хранилище для различных типов данных.*

Категория сравнения	СУБД	БД
Скорость извлечения данных	извлечение данных происходит **быстро**, т. к. компьютерная система учавствует в системе управления базами данных.	извлечение данных может происходить очень **медленно**, т.к. базы данных можно обрабатывать вручную или с помощью компьютеров, когда SQL не используется для извлечения информации.
Хранение	все записи ведутся только на компьютере.	помимо компьютера, могут поддерживаться в физических бухгалтерских книгах, книгах или документах.
Извлечение данных	с помощью запросов, написанных на SQL.	вручную, с помощью запросов или с помощью программ (C, C++< Java и т. д.).
Данные	управляет данными и манипулирует ими.	хранятся в базах данных.
Доступ	предназначена для большого количества людей, которые могут получить доступ к данным одновременно.	не предназначены для большого количества людей, которые могут одновременно получать доступ к данным, а скорее для очень небольшого числа людей (желательно нескольких человек), которые получают доступ к данным в разное время.
Резервное копирование и восставновление	гарантирует, что данные всегда будут доступны, даже после сбоев системы.	не гарантируют, что данные будут доступны после возникновения сбоя.
Манипулирование данными	за один раз может быть изменено большое количество информации (так как ее могут искользовать одновременно много пользователей).	за один раз может быть изменено небольшое количество информации.

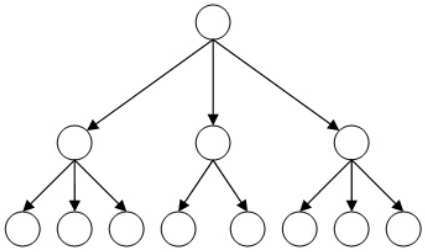
Виды баз данных

Существует огромное количество разновидностей баз данных, различающихся по различным критериям.

По модели данных

- **Иерархические.** Каждый объект, при таком хранение информации, представляется в виде определенной сущности, то есть у этой сущности могут быть дочерние элементы, родительские элементы, а у тех дочерних могут быть еще дочерние элементы, но есть один объект, с которого все начинается. Получается своеобразное дерево.

  Примером иерархической базы данных может быть документ в формате XML или файловая система компьютера.

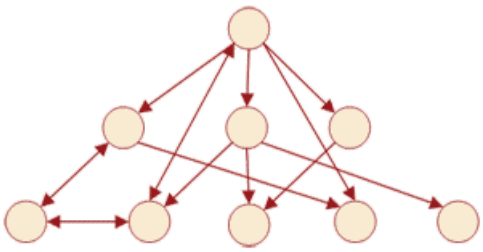


- **Объектные** — это модель работы с объектными данными. Такая модель баз данных, несмотря на то, что она существует уже много лет, считается новой. И её создание открывает большие перспективы, в связи с тем, что использование объектной модели баз данных легко воспринимается пользователем, так как создается высокий уровень абстракции. Объектная модель идеально подходит для трактовки такого рода объектных данных как изображение, музыка, видео, разного вида текст.
- **Объектно-ориентированные** — база данных, в которой данные моделируются в виде объектов, их атрибутов, методов и классов. Они обычно рекомендованы для тех случаев, когда требуется высокопроизводительная обработка данных, имеющих сложную структуру.
- **Объектно-реляционные.** Объединяют в себе черты реляционной и объектной моделей. Их возникновение объясняется тем, что реляционные базы данных хорошо работают со встроенными типами данных и гораздо хуже — с пользовательскими, нестандартными. Когда появляется новый важный тип данных, приходится либо включать его поддержку в СУБД, либо заставлять программиста самостоятельно управлять данными в приложении.
- **Реляционные.** Содержат перечень объектов одного типа, т.е. объектов с одинаковым набором свойств. Такую базу удобно представлять в виде

двумерной таблицы (или, чаще всего, нескольких связанных между собой таблиц).

Примером такой таблицы может служить БД «Учащиеся», представляющая собой перечень объектов (учеников), каждый из которых имеет фамилию, имя, отчество, дату рождения, класс, номер личного дела и др.

- **Сетевые.** Являются своеобразной модификацией иерархических баз данных. Если Вы внимательно смотрели на изображение выше, то наверняка обратили внимание, что к каждому нижнему элементу идет только одна стрелочка от верхнего элемента. То есть у иерархических баз данных у каждого дочернего элемента может быть только один потомок. Сетевые базы данных отличаются от иерархических тем, что у дочернего элемента может быть несколько предков, то есть элементов стоящих выше него. Для большей наглядности и понимания структуры сетевых баз данных обратите внимание на изображение:



- **Функциональные.** Используются для решения аналитических задач: финансовое моделирование и управление производительностью. Функциональная база данных или функциональная модель отличается от реляционной модели. Функциональная модель также отличается от других аналогично названных концепций, включая модель функциональной базы данных DAPLEX и базы данных функциональных языков.

По содержимому

- географическими;
- историческими;
- научными;
- мультимедийными;
- клиентская.

По степени распределённости

- **Централизованная или сосредоточенная.** Полностью поддерживается на одном компьютере.
- **Распределённая.** БД, составные части которой размещаются в различных узлах компьютерной сети в соответствии с каким-либо критерием.
- **Неоднородная.** Её фрагменты в разных узлах сети поддерживаются средствами более одной СУБД.
- **Однородная.** Её фрагментыв разных узлах сети поддерживаются средствами одной и той же СУБД.
- **Фрагментированная или секционированная.** Методом распределения данных является фрагментирование (партиционирование, секционирование), вертикальное или горизонтальное.
- **Тиражированная.** Методом распределения данных является тиражирование.

По среде физического хранения

- **БД во вторичной памяти (традиционные).** Средой постоянного хранения является периферийная энергонезависимая память (вторичная память) — это, как правило, жёсткий диск. В оперативную память СУБД помещает лишь кэш и данные для текущей обработки.
- **БД в оперативной памяти (*in-memory databases*).** Все данные находятся в оперативной памяти.
- **БД в третичной памяти (*tertiary databases*).** Средой постоянного хранения является отсоединяемое от сервера устройство массового хранения (третичная память), как правило, на основе магнитных лент или оптических дисков. Во вторичной памяти сервера хранится лишь каталог данных третичной памяти, файловый кэш и данные для текущей обработки; загрузка же самих данных требует специальной процедуры.

Типы баз данных

Oracle

— это объектно-реляционная СУБД (система управления базами данных), созданная компанией Oracle. К ее преимуществам можно отнемти то, что она имеет быструю установку и натсройку, возможность расширять функционал,

практичность и надежность. Oracle используется крупными компаниями, т.к. лицензия стоит дорого.

MySQL

— это система баз данных с открытым исходным кодом.

— это быстрая и простая в использовании СУБД, которая используется для разработки различных небольших и крупных приложений.

Он широко используется различными приложениями, такими как Joomla, WordPress, Drupal и многими другими. MySQL популярен благодаря различным возможностям.

Microsoft SQL Server

— система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Легко интегрируется с другими продуктами Microsoft, удобна в использовании, но потребляет много ресурсов, а лицензия стоит дорого.

PostgreSQL

— объектно-реляционная СУБД. Это значит, что она поддерживает и объектный, и реляционный подход. Поддержка множества типов данных. Еще одна **особенность PostgreSQL** — поддержка большого количества типов записи информации.

Apache Cassandra

— это нереляционная отказоустойчивая распределенная СУБД, рассчитанная на создание высокомасштабируемых и надёжных хранилищ огромных массивов данных, представленных в виде хэша. Система хранит данные по модели семейства столбцов и “ключ-значение”, расплередяет данные в несколько дата-центров и легко масштабируется при увеличении объема памяти.

SQL

***SQL* (аббр. от англ. *Structured Query Language* — «язык структурированных запросов»)** — это структурированный язык запросов, созданный для того, чтобы получать из базы данных необходимую

информацию. Если описать схему работы SQL простыми словами, то специалист формирует запрос и направляет его в базу. Та в свою очередь обрабатывает эту информацию, «понимает», что именно нужно специалисту, и отправляет ответ.

Почти все *реляционные базы данных* используют **SQL**.

Независимо от того, какой язык программирования используют для реализации процессов в компании (Python, C, C++), SQL все равно нужен для того, чтобы извлекать необходимую информацию из СУБД.

Основные преимущества SQL:

- **точность** — можно не хранить избыточные данные;
- **гибкость** — даже самые сложные запросы легко выполнить;
- **масштабируемость** — с одной БД могут работать множество пользователей;
- **безопасность** — доступ к данным в таблицах есть только у определенных пользователей.

На языке SQL выражаются все действия, которые можно провести с данными: от записи и чтения данных, до администрирования самого сервера СУБД.

Для повседневной работы совсем не обязательно знать весь этот язык; достаточно ознакомиться лишь с основными понятиями синтаксиса и ключевыми словами. Кроме того, SQL очень простой язык по своей структуре, поэтому его освоение не составит большого труда.

С помощью SQL можно не только добавлять и читать данные, но и:

- удалять и обновлять записи в таблицах;
- создавать и редактировать сами таблицы;
- производить операции над данными: считать сумму, получать самое большое или малое значение, и так далее;
- настраивать работу сервера СУБД.

Описание базовых запросов SQL

- **CRUD (создание, чтение, обновление, удаление)** — это аббревиатура, обозначающая четыре функции, которые мы используем для реализации

приложений постоянного хранения и приложений реляционных баз данных, включая Oracle Database, Microsoft SQL Server и MySQL.

В таблице ниже показано, что означает каждая операция CRUD:

Письмо	Операция	Функция
C	Создавать	Вставлять
p	Читать	Выбирать
U	Обновлять	Редактировать
D	Удалить	Удалить

Почему CRUD так важен?

CRUD постоянно используется для всего, что связано с базами данных и проектированием баз данных. Разработчики программного обеспечения ничего не могут сделать без операций CRUD. Например, при разработке веб-сайтов используется REST (передача репрезентативного состояния), который является надмножеством CRUD, используемого для ресурсов HTTP.

С другой стороны, CRUD не менее важен для конечных пользователей. Без него такие вещи, как регистрация на веб-сайтах, создание блогов или закладок, были бы невозможны. Большинство приложений, которые мы используем, позволяют нам добавлять или создавать новые записи, искать существующие, вносить в них изменения или удалять их.

CRUD предлагает множество преимуществ, в том числе:

- Это облегчает контроль безопасности, удовлетворяя различные требования доступа.
- Он упрощает и упрощает разработку приложений, делая их более масштабируемыми.
- Он имеет лучшую производительность по сравнению со специальными операторами SQL.

- **SELECT**

  Наиболее используемым, но и самым сложным оператором является оператор выборки SELECT. Он позволяет производить выборку данных из таблиц и преобразовывать к нужному виду полученные результаты.

Результатом выполнения оператора `SELECT` является таблица. К этой таблице может быть снова применен оператор `SELECT` и т.д., то есть такие операторы могут быть вложены друг в друга. Вложенные операторы `SELECT` называют подзапросами.

Синтаксис оператора `SELECT` использует следующие основные предложения:

```
SELECT <список столбцов>
FROM <список таблиц>
[WHERE <условие выбора строк>]
[GROUP BY <условие группировки>]
[HAVING <условие выбора групп>]
[ORDER BY <условие сортировки>]
```

Кратко пояснить смысл предложений оператора `SELECT` можно следующим образом:

- `SELECT` - выбрать данные из указанных столбцов и (если необходимо) выполнить перед выводом их преобразование в соответствии с указанными выражениями и (или) функциями
- `FROM` - из перечисленных таблиц, в которых расположены эти столбцы
- `WHERE` - где строки из указанных таблиц должны удовлетворять указанному перечню условий отбора строк
- `GROUP BY` - группируя по указанному перечню столбцов с тем, чтобы получить для каждой группы единственное значение
- `HAVING` - имея в результате лишь те группы, которые удовлетворяют указанному перечню условий отбора групп
- `ORDER BY` - сортируя по указанному перечню столбцов

Как видно из синтаксиса рассматриваемого оператора, обязательными являются только два первых предложения: `SELECT` и `FROM`.

DB Testing

Зачем тестировщику нужно уметь работать с базами данных?

Для того, чтобы:

- **Создавать начальные данные.** В реальной системе много отношений между моделями, чтобы вставить хотя бы одну строку в целевую таблицу необходимо заполнить множество связанных таблиц. Например, товар (Good) может ссылаться на производителя (Manufacturer), который в свою очередь ссылается на страну (Country).

  Чтобы упростить дальнейшее создание тестовых сценариев, необходимо создать минимальный набор общих для системы данных.

- **Подготовка окружения.** Тестовое окружение в первую очередь это база данных, также это могут быть синглтоны и статические переменные. Лучше собрать все эти операции в одном месте и запускать перед каждым тестом.

  Возможность задать статические переменные и синглтоны особенно важна при тестировании legacy кода, где не так-то просто поменять архитектуру — но есть острая необходимость в тестировании. Разделение настройку окружения на несколько методов позволяет подготавливать окружение индивидуального для каждого теста. Например, в unit тестах не используется база и нет смысла очищать для них базу. Или у вас может быть необходимость подготовить различное окружение для разных состояний системы (авторизованный и неавторизованный пользователь).

- **Сравнить** реально существующие данные в базе с теми, которые отображаются в приложении, которое мы тестируем. Ведь отображение это дополнительная прослойка, плюс запрос в коде может быть реализован некорректно;

- **Создавать тестовый сценарий.** В тестах приходится делать много подготовительной работы, Arrange фаза теста самая ответственная и сложная. Поэтому желательно создавать хелперы, которые упростят этот процесс, сделают код более простым для чтения. Одним из удобных механизмов, может быть создание ModelBuilder, который создает сущности, сохраняет их в БД и возвращает экземпляры для дальнейшего использования.

- **Напрямую загружать свои тестовые данные в БД**, чтобы проверить какие-то сценарии. Например, если создание данных ещё не реализовано или для ускорения тестирования (быстрее добавить напрямую, чем через возможности приложения);

- **Удалять данные и редактировать** напрямую в БД.

- **Тестировать базы данных.** Тестирование БД включает в себя проверку достоверности данных, целостности данных, производительности, связанную с базой данных и тестирование процедур, триггеров и функций в баще данных.

Как при ручном тестировании, так и при автоматическом могут использоваться прямые запросы в БД. Именно поэтому нужно уметь с ними работать.

К тому же знание особенностей работы различных баз данных позволяет вывести тестирование на совершенно другой профессиональный уровень.

Стоит отметить ещё, что есть разработчики и тестировщики баз данных. А иногда, мы можем тестировать базу данных отдельно от всего приложения, как отдельный компонент системы